Multi-View Stereo 3D Edge Reconstruction

Andrea Bignoli Andrea Romanoni Matteo Matteucci
Politecnico di Milano
andrea.bignoli@mail.polimi.it andrea.romanoni@polimi.it matteo.matteucci@polimi.it

Abstract

This paper presents a novel method for the reconstruction of 3D edges in multi-view stereo scenarios. Previous research in the field typically relied on video sequences and limited the reconstruction process to either straight line-segments, or edge-points, i.e.*, 3D points that correspond to image edges. We instead propose a system, denoted as EdgeGraph3D, able to recover both straight and curved 3D edges from an unordered image sequence. A second contribution of this work is a graph-based representation for 2D edges that allows the identification of the most structurally significant edges detected in an image. We integrate EdgeGraph3D in a multi-view stereo reconstruction pipeline and analyze the benefits provided by 3D edges to the accuracy of the recovered surfaces. We evaluate the effectiveness of our approach on multiple datasets from two different collections in the multi-view stereo literature. Experimental results demonstrate the ability of EdgeGraph3D to work in presence of strong illumination changes and reflections, which are usually detrimental to the effectiveness of classical photometric reconstruction systems.*

1. Introduction

Reconstructing the 3D shape of a scene captured by a set of images represents a long-standing problem faced by the computer vision community. Structure from Motion methods address the simultaneous estimation of camera positions and orientations together with a point-based representation of the environment [27, 15, 17, 20]. Multi-View Stereo algorithms usually bootstrap from such estimations to recover a mesh-based dense reconstruction.

State-of-the-art mesh-based algorithms [26, 12, 11, 18] are initialized through Delaunay-based space carving algorithms such as [24, 13, 9, 19] which estimate a mesh from the structure from motion points or from dense point clouds computed through depth maps. The authors of [19] showed that the Delaunay triangulation built upon 3D edge-points, *i.e.*, points belonging to 3D edges, is able to represent the shape of the environment better than using 3D points reconstructed from classical 2D features. The usage of 3D edges or 3D edge-points presents two significant benefits: they are robust to significant illumination changes that can negatively affect standard photometric-based depth maps estimation, and they are a compact representation of the salient part of a scene, i.e., they avoid redundancies along flat surfaces.

The reconstruction of 3D edges can be performed by matching directly their 2D observations across a sequence of images. This is a challenging task since corresponding edges often cannot be matched just on the basis of their geometric parameters. In literature, edge reconstruction is often limited to the reconstruction of line-segments, *i.e.*, straight edges, and existing approaches rely their estimation on video sequences. Only Hofer *et al*. [8] with Line3D++ proposed an approach to estimate 3D segments in a Multi-View Stereo scenario. This method, however, is not able to recover curved edges.

In this paper we propose a novel algorithm for the reconstruction of 3D edges, both straight and curved, from an unordered set of images. Furthermore, we illustrate how the points belonging to 3D edges can significantly improve the appearance and the accuracy of the 3D models reconstructed from sparse point clouds, using the algorithm proposed by [13] and improved by [19]. We tested our algorithms on the fountain-p11 dataset provided in the EPFL Multi-View Stereo collection [21] and on the recent DTU dataset [10].

In Section 2 we overview the state-of-the art of 3D edge reconstruction. In Section 3 we show the 2D edge representation we use in our algorithm. In Section 4 we describe the proposed method to reconstruct 3D edges and introduce the EdgeGraph3D system. In Section 5 we discuss the results of our approach on two well-known datasets. In Section 6 we conclude the paper and we illustrate some possible future research directions.

2. Related works

In literature, a limited amount of works address directly the issue of edge reconstruction, and usually they recover only long straight edges. Tian *et al*. [23] track 2D points

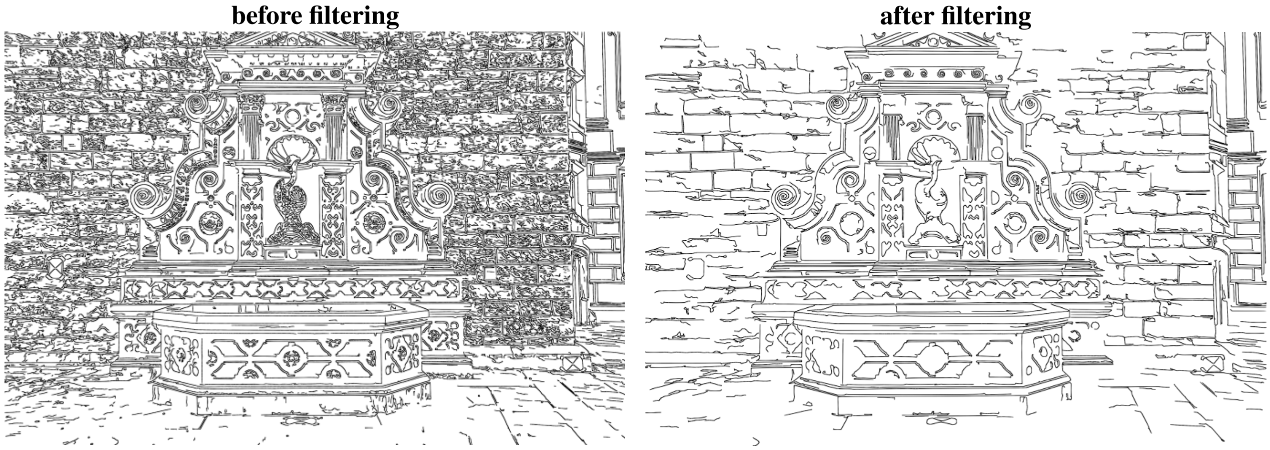

Figure 1. Edge-graph filtering: (left) original edge-graph, generated from an edge-image produced by the algorithm presented in [14] (right) filtered edge-graph, in which non-structural edges have been removed

belonging to images edges along a video sequence and estimate their 3D position, then they recover the 3D edges that connect these 3D points. Zhang and Baltsavias [28] propose an algorithm for the reconstruction of road edges from aerial images. They match only straight edges, since these are prevalent in urban environments, by using epipolar geometry.

Edges have been also integrated in structure from motion pipelines as elements to robustly recover or optimize the camera poses. In [22], the authors propose a structure from motion algorithm based on-line segments correspondences in an image sequence. Instead of classical point-based reprojection error measure, they optimize a non-linear objective function that measures the total squared distance between the observed edge segments and the projections of the reconstructed lines on the image plane. Ansar *et al.* [1] propose the mathematical foundations of a camera pose estimation systems able to use both points and lines for real-time camera pose estimation using linear optimization, while [6] proposes a monocular Simultaneous Localization and Mapping (SLAM) algorithm which adopts as landmarks small edges, *i.e.*, the edgelets, instead of the classical point-based features. Also the latter algorithms are able to estimate 3D straight edges, while curved edges are neglected, moreover they rely on video sequences.

In [19], the authors estimate the 3D position of 2D points belonging to images edges, and they embedded them into a Delaunay triangulation to make the triangulation edges closer to the real 3D structure of the scene. In this case even points on curved edges are taken into account, but the actual 3D edges are not explicitly reconstructed. Similarly, Bodis *et al.* [3] present a reconstruction algorithm for large-scale multi-view stereo, able to produce meshes that are consistent with the bidimensional edges of the input images; they enforce the Delaunay Triangulation, employed in the reconstruction, to be properly divided along edges. While the purpose of the system is not the actual reconstruction of the 3D edges, they further show the benefits that edges offer to the reconstruction process by sharply defining the architectural elements in urban scenes.

This review shows that edges or the points belonging to edges are sometimes adopted to improve the robustness of 3D reconstruction, camera tracking or SLAM algorithms. However these techniques deal only with video sequences to simplify the process of edge matching and they focus only on straight edges estimation; a general approach to 3D edge reconstruction from unordered set of images is thus the novel contribution of this paper that is also beneficial for more complex dense 3D reconstruction algorithms.

3. 2D Edge-Graphs

Decades of research in computer vision produced several edge detection algorithms. Many of the proposed techniques, such as [4], are able to describe both straight and curved edges through an image output, but they represent the edges only at pixel level. On the other hand, other approaches as [25], estimate line-segments, which describe edges with subpixel accuracy, but are not able to properly represent and detect curved edges. Here, we introduce an alternative representation for 2D edges, able to both represent curved edges and to reach sub-pixel accuracy. We propose to use an undirected graph, named *2D edge-graph*, in which nodes represent 2D points on an image, and connections between nodes indicate detected edges connecting their extremes. An additional benefit provided by the use of edge-graphs is the direct description of the connections between different edges detected in an image, which will be key to the techniques presented in Section 4.3.

We generate an edge-graph from the edge-images produced by the standard edge-detection algorithms presented

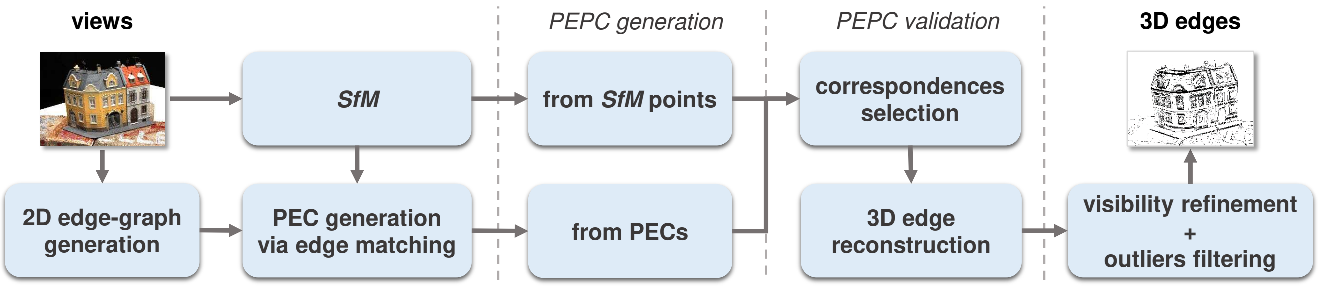

Figure 2. Proposed 3D edge reconstruction pipeline

in [14]. We assign a node to the center of each edge-pixel; then, for each pair of adjacent edge-pixels, we connect the corresponding nodes if it does not generate small, undesired, loops of length shorter than 4 *px*, *e.g.*, which do not represent meaningful connections between edges in the original edge-image. Since edges recovered using this process still follow the discretized structure described by the original edge-image, we apply a *polyline smoothing* step. A polyline is a sequence of edges in the graph, in which all intermediate nodes have exactly two connections. Polyline smoothing is a process by which the original polyline is transformed so that: **i)** the extremes of the polyline are left unaltered **ii)** the sequence of intermediate nodes is modified to the shortest sequence guaranteeing that a distance no greater than 1 pixel separates the original polyline from the final one. We employed a variant of the Douglas-Peucker [5] algorithm to achieve this result. Using this technique, it is possible to obtain a suitable representation for edges in a scene, at subpixel-accuracy.

Many standard edge detection algorithms generate a significant number of edges that do not correspond to structural elements in the scene. To filter them out, we propose an edge-graph filtering step which retains only long polylines composed by line-segments without significant sharp variations in direction. Let us define:

Definition Regular length The *regular length* of a polyline is the length of the longest interval of connected line-segments for which each angle between consecutive elements is not greater than a fixed threshold α_R (*e.g.*, $\alpha_R \approx 20°$).

For each 2D edge-graph we rank all its polylines according to their regular length and we compute l_R^* as the shortest length among the top 10% of these polylines. We then filter out all the connected components in the graph that do not contain a polyline with regular length greater than l_R^*. In Figure 1, we show that structural edges are preserved, while irrelevant edges are almost completely filtered out.

4. The EdgeGraph3D system

EdgeGraph3D is able to reconstruct three-dimensional edges from their observations in the input views. Curved edges are represented as 3D polylines, *i.e.*, a sequence of straight 3D line-segments connected to each other. The inputs of our method are the 2D edge graphs computed for each image, the camera calibrations and an initial set of 3D points estimated through structure from motion. To understand the key idea behind the proposed algorithm, let us assume a 3D edge-point and its 2D observations on a subset of images are provided; these observations likely lie on 2D edges. We simultaneously follow such 2D edges on all the images involved, generating a sequence of corresponding 2D edge-points that identify new 3D edge-points. The sequence of the recovered 3D edge-points defines a 3D polyline representing the reconstructed 3D edge.

4.1. System Overview

In Figure 2 we illustrate the full pipeline of EdgeGraph3D. From the input images we compute the camera calibration via SfM. For each image we define the corresponding 2D Edge-Graph presented in the previous section that we use to match edges on multiple views and to define

Definition Potential edge correspondence (PEC) A *PEC* is a set of 2D polylines (*i.e.*, image edges), on multiple views, that are considered projections the same 3D edges.

In Section 4.3 we illustrate how, for each image 2D edge-point, we exploit the epipolar constraint to generate

Definition Potential edge-point correspondence (PEPC) A *PEPC* is a set of possibly corresponding 2D edge-points on multiple views, which may even contain multiple points on the same view, from which it may be possible to generate a new 3D edge-point.

We bound the number of *PEPC*, by relying on the spatial information carried by the *SfM* 3D points and on the edge matches collected in the *PEC* s. In Section 4.4, we validate each *PEPC*, while reconstructing 3D edges. Finally in Section 4.4.3 we illustrate how we improve the visibility

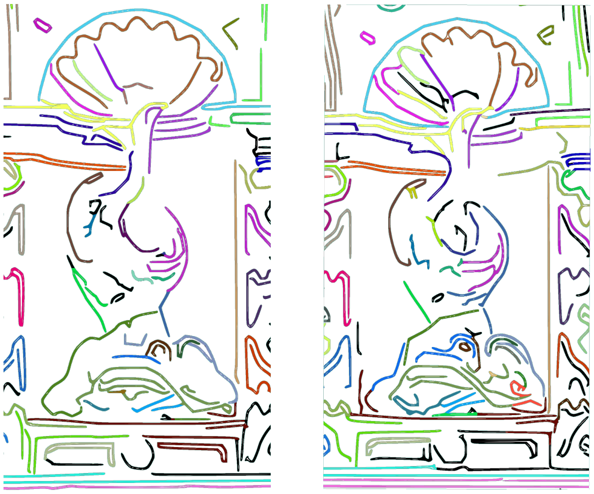

Figure 3. Visualization of the output of the edge matching procedure on two 2D edge-graphs of the fountain-P11 [21] dataset. Polylines with the same color belong to the same *PEC*. This sample visually show the effectiveness of the approach, regardless of the incompleteness of the detected edges on both images.

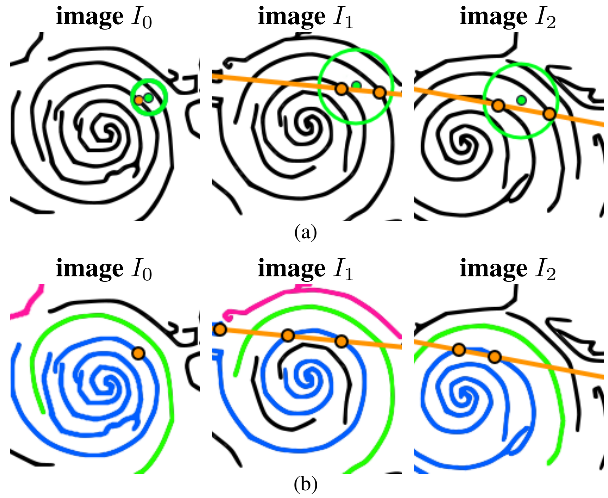

Figure 4. Generation of *PEPCs*: (a) searching a new edge-point (in orange) in the vicinity of a known 3D point (in green) (b) searching a new edge-point using polyline matches, represented by edges shown in the same color on different images. A new orange 2D edge-point on the blue match is selected as target.

information associated to a 3D edge and how we remove outliers.

4.2. Edge Matching and *PEC* generation

To compute *PECs* let us consider a pair of 2D polylines γ_i and γ_j on images I_i and I_j; if a pair of polylines shares a significant amount of nearby 3D points, they must occupy nearby locations in the 3D space, and are potentially associated to the same 3D edges. Therefore we define a similarity measure $\mathrm{s}\left(\gamma_i, \gamma_j\right)$ for γ_i and γ_j as:

$$\mathrm{s}\left(\gamma_i, \gamma_j\right) = \frac{\sum_{p \in P^{\gamma_i} \cap P^{\gamma_j}} w_p}{\sum_{p \in P^{\gamma_i} \cup P^{\gamma_j}} w_p}, \qquad (1)$$

where P^{γ_i} is the list of 3D points visible on I_i, that lie within a distance of d_s from γ_i, and the weight w_p of a point p is defined as the inverse of the average number of polylines close to the reprojections of p, where p is visible.

Then, we build a *polyline similarity graph* as an undirected weighted graph, in which nodes represent different polylines on different images, and the weight of each edge is equal to the similarity of the polylines associated to its extremes. We then use the community detection algorithm in [2], on the polyline similarity graph. Communities are subsets of nodes of a graph that are densely interconnected, hence in our case they are subsets of polylines, on multiple views, with high degree of similarity, *i.e.*, they represent *PECs*, as Figure 4 shows.

4.3. *PEPC* generation

To present the principle that inspires the two strategies for *PEPCs* generation presented in this section, let us consider a 2D edge-point x_o in the 2D edge-graph of image I_o. To recover the corresponding 3D edge-point x we identify the potential 2D edge-points correspondences on other views through epipolar geometry. The correspondence x_i on a second image $I_i \neq I_o$ must lie on the epipolar line l_i generated on I_i by x_o. Since we assume that the correspondence x_i is a 2D edge-point, we generate a finite set of potential correspondences $H_i = \{h_{i1}, h_{i2}, \dots h_{im}\}$ by intersecting the 2D edge-graph associated to I_i with the epipolar line l_i. We repeat this process for all other views where the new potential 3D edge-point may be visible. The cardinality of the sets of potential correspondences H_i is however generally too high to search for the correct correspondences on all views in acceptable computation time. Therefore, we propose two approaches to limit the set of potential correspondences on each view.

4.3.1 From *SfM* points

In the first approach (Figure 4(a)) we exploit the knowledge of a 3D point position p, recovered through structure from motion, and we constrain the search for a new 3D edge-point x in its neighborhood, in particular, to a sphere $\mathcal{S}_O$ centered in p with radius r_O. Given an image I_o where p projects on a location p_o, we search the initial 2D edge-point x_o within the projection of a sphere $\mathcal{S}_I$ centered in p with a radius $r_I < r_O$. This projection is an elliptic conic section that we approximate with a circle $\mathcal{O}_o$ centered in p_o (green circle in image I_o of Figure 4(a)), of radius $r_o = r_I \frac{||c_o - p||}{f}$, where c_o represents the center of camera C_o that produced I_o, and f is the focal length. Then, for each polyline passing through $\mathcal{O}_o$, we select the 2D edge-point closest to p (orange point in image I_o of Figure 4(a)) as the initial edge-points for which we aim to find 2D to 2D correspondences on other views.

For each image $I_i \neq I_o$ where p projects, we look for correspondences near the projection p_i of p on I_i (orange point in images I_1 and I_2 of Figure 4(a)). In particular, we constrain the search of correspondences within circle $\mathcal{O}_i$, centered in p_i, of radius $r_i = r_O \frac{||c_i - p||}{f}$, which approximates the projection of sphere $\mathcal{S}_O$ on I_i (green circles in images I_1 and I_2 of Figure 4(a)). Since the correspondences of x_o on I_i are bound to lie on the epipolar line l_o, we combine both constraint looking for intersections between l_o and the edge-graph in $\mathcal{O}_i$, to determine the set of potential correspondences H_i of x_o on image I_i. Repeating the same process on multiple views generates a new *PEPC*.

4.3.2 From *PECs*

The second approach (Figure 4(b)) makes use of *PECs* generated using technique presented in Section 4.2. Formally, a *PEC* is a set $M = \{M_1, M_2, \ldots M_N\}$, which associates to each image I_i of the N views observing the scene, a set M_i of polylines of the corresponding 2D edge-graph involved in the match.

To generate the *PEPCs*, let us consider one of the views with a nonempty set of matched polylines set as the initial view I_o. We select an edge-point x_o on one of the matched polylines. Correspondences on each view I_i can be identified by intersecting the matched polylines on I_i set with epipolar line l_o generated by x_o. Repeating the process for all the views produces a set of possibly corresponding 2D edge-points, *i.e.*, a *PEPC*. The process is repeated for different initial edge-points, obtained by sampling the polylines on an initial view at fixed intervals, to generate multiple *PEPCs* from a single *PEC*.

4.4. *PEPC* validation and 3D edge generation

Each *PEPCs* generated with the techniques presented so far associates a set of potential 2D correspondences on images $I_i \neq I_o$ to a 2D edge-point x_o on I_o. Given a *PEPC* we define:

Definition PEPC-Selection A *PEPC-selection* is a subset of the *PEPC* 2D correspondences such that each image has at most one correspondence.

In Section 4.4.1 we explain how we choose among the vast set of potential selections to recover the 3D edge-point x that generated x_o. In Section 4.4.2 we present the technique we use to recover from x the corresponding 3D edge. In Section 4.4.3, we refine the visibility of the generated edges, and we remove outliers. Figure 5 illustrates the complete *PEPC* validation and 3D edge reconstruction pipeline.

4.4.1 Correspondences selection

A correct PEPC-selection retains, for each view, the one, if it exists, associated to the initial 2D edge-point x_o on the initial view I_o used to generate the *PEPC*. The identification of the correct *PEPC*-selection is therefore a combinatorial problem defined on the search space of all possible selections of edge-point correspondences. A PEPC-selection of 2D edge-point correspondences requires at least three observations to provide minimal geometrical evidence that the correspondences identify the same 3D point, by three-view triangulation. The number of potentially acceptable selections, with at least three views, is therefore:

$$\underbrace{\prod_{\substack{H_i \in H, \\ H_i \neq H_o}} (|H_i| + 1)}_{\substack{\text{all combinations} \\ \text{that include } x_o}} - \underbrace{\overbrace{\sum_{\substack{H_i \in H, \\ H_i \neq H_o}} |H_i|}^{\substack{\text{pairs of the form} \\ (x_o, h_{ia})}} - \overbrace{1}^{\substack{\text{only } x_o \\ \text{is selected}}}}_{\substack{\text{invalid combinations with} \\ \text{less than 3 selected correspondences}}}, \quad (2)$$

where H_i is the set of correspondences in the *PEPC* on view I_i, and H is the set of all H_i. To reduce the size of the search space, we initially limit the selection problem to only three views, of which one is the initial view I_o. The other two views I_i and I_j can be chosen arbitrarily. All the potential selections on the three views, amounting to $|H_i| * |H_j|$, can be independently checked for correctness.

A PEPC-selection is not acceptable if it is not possible to generate a 3D point from it, through multi-view triangulation, with a small maximum reprojection error of ϵ (*e.g.*, $\epsilon \approx 2 - 3$ *px*). Due to inaccuracies, however, incorrect PEPC-selections can satisfy the geometrical constraint imposed by the triangulation, hence to be considered valid a selection must: (i) triangulate to a valid 3D point (ii) generate a valid 3D edge, as presented in Section 4.4.2 . We only accept a PEPC-selection if it is the only one in the *PEPC* that respects the above conditions.

4.4.2 3D Edge reconstruction

Given an initial PEPC-selection and the triangulated 3D edge-point we check whether it is possible to follow the 2D edge-graphs among images to generate a sequence of corresponding 2D edge-points. Starting from the initial edge-point x_o^s generated by a 3D edge-point x^s on the first view I_o, it is possible to follow the 2D polyline in two different directions. The first step match the different directions of the polylines on only three images, *i.e.*, I_0, I_1 and I_2.

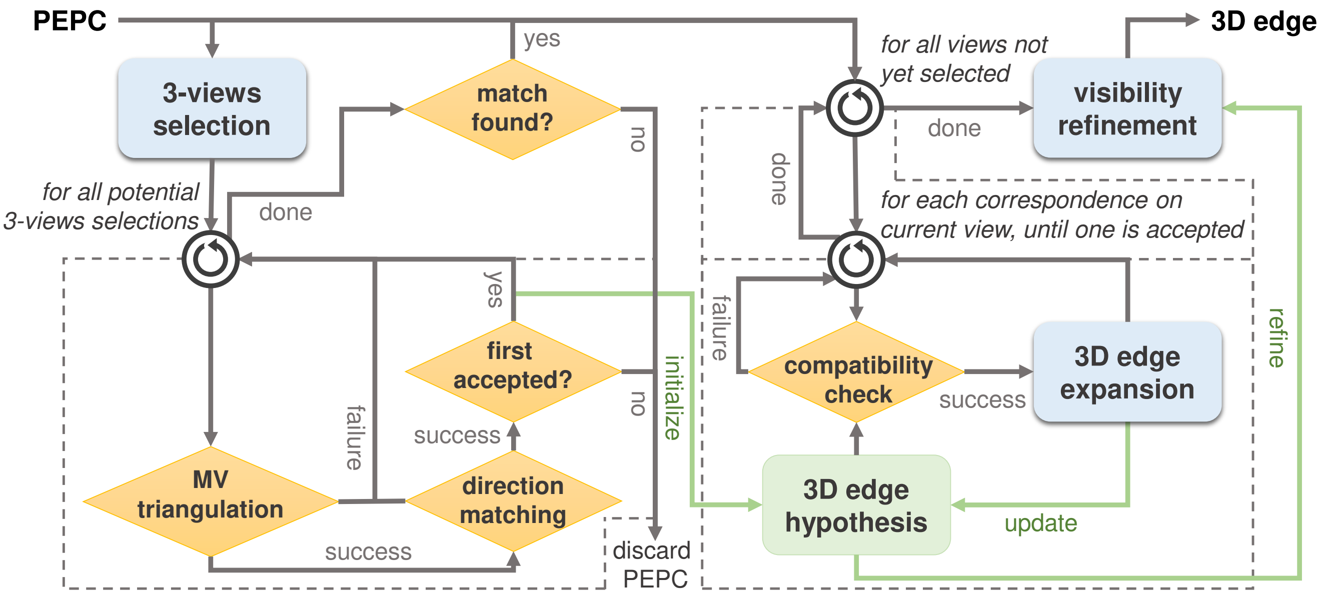

Figure 5. *PEPC* validation pipeline

We first move along the corresponding polyline in the direction d_0, by a small fixed length $l_d \approx 10\ px$, and select a new edge-point x_o^a on I_o. The sampling interval l_d has been chosen, without tuning, to closely follow the direction changes of a polyline, without generating irrelevant edge-points. The correspondences of x_o^a on the other view I_i can be identified by tracing the epipolar line on I_i and selecting the first intersection x_i^a between that line and the corresponding polyline on I_i, starting from the initial edge-point observation x_i^s, and moving towards the direction d_i. We apply this process on both views 1 and 2, and we verify that the two edge-point correspondences, x_i^a and x_j^a together with x_o^a triangulate to a valid 3D point. If this operation is successful, the direction is considered valid. New edge-points can be recursively found following the directions (d_0, d_1, d_2) and the opposite $(-d_0, -d_1, -d_2)$ until a failure, either in finding correspondences or in the multi-view triangulation, occurs. Note that the extent of the movement between edge-point samples on the first view controls for the degree of approximation of curved edges.

The above process is the initial step in the definition of a new 3D edge. If a solution is accepted for three views, *i.e.*, it is the only valid selection, other potential 2D edge-point correspondences in the *PEPC* on additional views, excluded in the initial selection, should still be integrated if compatible with the current 3D edge. We consider a new potential 2D to 3D edge-point observation h_{k1} for the initial 3D edge-point x^s, on a new view I_k. The first step in validating the new observation is checking whether it is compatible, by multi-view triangulation, with the current bidimensional observations of x^s and that it is possible to match directions between the current 3D polyline-edge, and the 2D polyline on I_k. Using the new observations, we can further extend the current 3D edge, and improve its accuracy.

4.4.3 Visibility refinement and outliers filtering

Once a 3D polyline-edge γ is generated, we can optimize its visibility information by checking whether the 3D edge is visible in a view that was not considered so far. Let us consider I_i, on which no observation of the 3D polyline γ has been found. New bidimensional observations of polyline γ, if existent on I_i, are expected to be near the projection of γ on I_i. In the proposed system, we look for new 2D edge-point observations for each of the 3D edge-points that define γ. Let us consider $x^a \in \gamma$, and its projection $\tilde{x^a}_i$ on I_i. The goal is finding, if existent, a new edge-point observation x_i^a of x^a on I_i. We look for polylines within a distance d_v from $\tilde{x^a}_i$. If a single polyline γ_i is found, we select the edge-point on γ_i closest to $\tilde{x^a}_i$ as the new potential 2D observation x_i^a of x^a. By multi-view triangulation we verify the compatibility of x_i^a with the current observations set of x^a. If this check is successful, we can verify that the 2D polyline γ_i is compatible, in the vicinity of x_i^a, with the 3D polyline γ. This can be done by matching both directions of γ from the initial point $x^a \in \gamma$, with the two directions on γ_i starting from x_i^a, using techniques analogous to the ones presented in Section 4.4.2. This process is applied to all edge-points of γ that have not been observed on I_i yet, and can be repeated for every view I_i to ensure that all potential observations of the 3D polyline γ are correctly identified.

Table 1. Comparison between EdgeGraph3D and OpenMVG.

		num.	point cloud			mesh		
		points	MAE	RMSE	σ	MAE	RMSE	σ
fountain-P11	OpenMVG	5570	**8.433**	**9.603**	**12.78**	88.94	209.3	189.4
	EdgeGraph3D	**41725**	12.19	15.98	20.10	**64.58**	**159.3**	**145.6**
DTU-006	OpenMVG	5903	**0.477**	**0.938**	**1.052**	4.077	11.53	10.79
	EdgeGraph3D	**47927**	0.542	1.230	1.344	**2.805**	**8.497**	**8.021**
DTU-023	OpenMVG	9651	0.826	**1.886**	2.059	4.585	9.223	8.003
	EdgeGraph3D	**97770**	**0.825**	1.911	**0.020**	**3.207**	**7.898**	**7.218**
DTU-028	OpenMVG	5008	1.961	3.607	4.106	20.24	54.07	50.14
	EdgeGraph3D	**46220**	**1.013**	**2.766**	**2.946**	**13.22**	**35.74**	**33.21**
DTU-037	OpenMVG	4321	**1.326**	**1.830**	**2.260**	**23.36**	40.21	32.72
	EdgeGraph3D	**38577**	1.478	2.372	2.795	25.32	**40.12**	**31.12**
DTU-098	OpenMVG	2091	4.658	9.501	10.58	24.89	55.32	49.40
	EdgeGraph3D	**26575**	**3.831**	**8.449**	**9.277**	**5.907**	**18.42**	**17.45**
DTU-118	OpenMVG	1839	**2.770**	**7.311**	**7.818**	19.93	39.49	34.10
	EdgeGraph3D	**14611**	3.065	7.894	8.468	**7.806**	**21.06**	**19.55**
average variation			+2.8%	+13.7%	+10.6%	**-36.0%**	**-30.3%**	**-29.2%**

Finally, we consider polylines with a low amount of observations to be likely outliers, hence we filter them out. The minimum amount of observations k_v is computed as $k_v = max\left(4, \frac{v_M}{2} + 1\right)$, where v_M is the median number of observations for all the 3D edge-points recovered by the system. The output of this final step is a set of accurate 3D polyline-edges that can properly represent even curved edges.

5. Experimental Results

We evaluate the results obtained by EdgeGraph3D on the fountain-P11 sequence of the EPFL dataset [21] and on 6 sequences of the DTU dataset [10] by reconstructing the 3D edges, extracting a point cloud by finely sampling them, used to reconstruct a mesh using the algorithm in [19] which is then compared with the ground truth. All the tests have been conducted on a Intel i5-3570K quad-core processor (3.80 GHz frequency, 6 MB smart cache) and 8 GB of DDR3 RAM. The values of the parameters of the algorithms presented in Section 4, such as the maximum reprojection error of ϵ, have been chosen to properly represent the geometrical properties associated with them and have not been subject to tuning in our experiments.

Our algorithm bootstraps from the SfM point cloud generated by OpenMVG [16], which provides very accurate points; therefore we compare the point cloud sampled from the 3D edges, against those estimated by OpenMVG by means of the CloudCompare software [7]. In Table 1 we list the Mean Absolute Errors (MAE), the Root Mean Squared Errors (RMSE) and the variance of the errors (σ). As expected, we significantly increase the number of reconstructed points (by one order of magnitude); despite our algorithm reconstructed full 3D edges, the accuracy of the reconstructed points remained close to the accuracy of the OpenMVG point cloud, which are easier to estimate, and in some cases this accuracy is even improved. As Figure 6 shows, the proposed algorithm is able to recover structural elements that may remain completely undetected by the standard SfM process. For instance in DTU-006, we are able to reconstruct edges of any inclination, recovering all the structural elements in the scenes; in the DTU-098 dataset, the high reflectivity of the metallic cans causes the SfM pipeline to fail in reconstructing a considerable portion of the surfaces, while the same areas are fully recovered by the proposed system.

Table 2. Comparison between meshes produced by EdgeGraph3D and Line3D++ [8].

		MAE	RMSE	σ
fountain-P11	Line3D++	101.8	272.5	252.7
	EdgeGraph3D	**64.59**	**159.4**	**145.7**
DTU-006	Line3D++	**1.792**	**6.552**	**6.302**
	EdgeGraph3D	2.805	8.497	8.021
DTU-023	Line3D++	4.778	10.06	8.851
	EdgeGraph3D	**3.207**	**7.898**	**7.218**
DTU-028	Line3D++	21.10	60.19	56.38
	EdgeGraph3D	**13.22**	**35.74**	**33.21**
DTU-037	Line3D++	**20.13**	**36.18**	**30.07**
	EdgeGraph3D	25.32	40.12	31.12
DTU-098	Line3D++	17.69	47.22	43.79
	EdgeGraph3D	**5.907**	**18.42**	**17.45**
DTU-118	Line3D++	9.498	**20.05**	**17.66**
	EdgeGraph3D	**7.806**	21.06	19.56
average variation		**-15.6%**	**-17.0%**	**-17.2%**

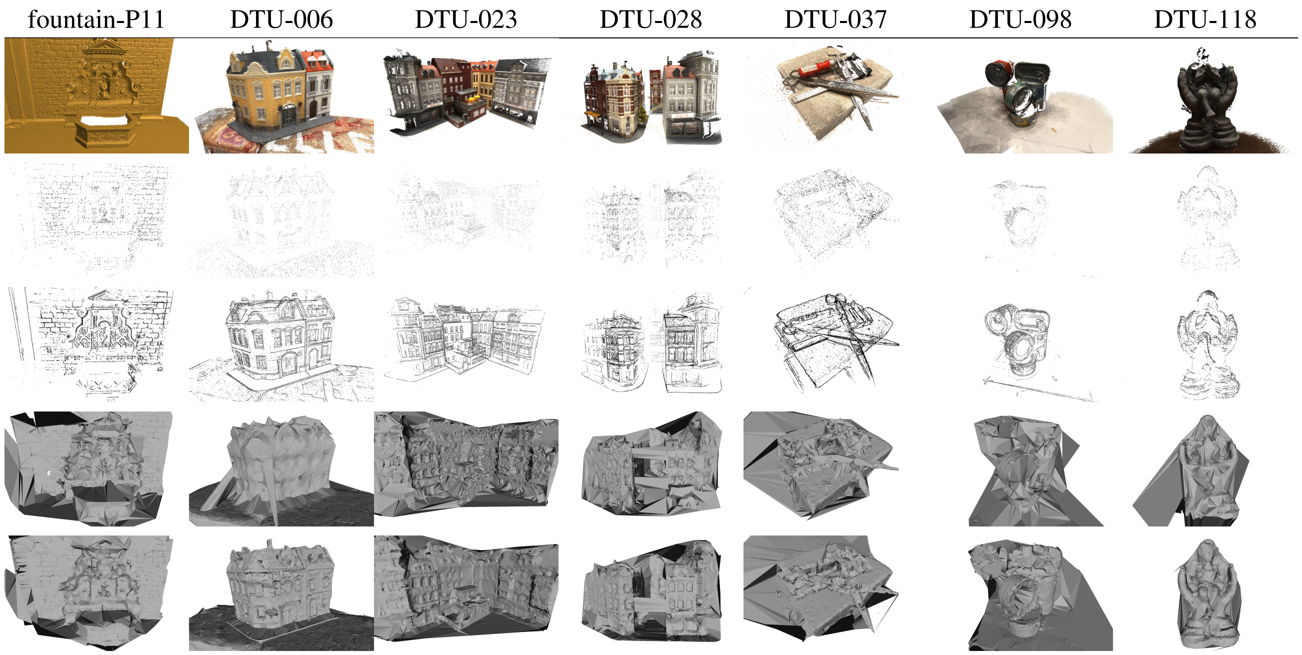

Figure 6. By row: ground truth, point cloud from OpenMVG, point cloud from our method, mesh from OpenMVG cloud, mesh from proposed cloud

Since the enhancement of 3D Delaunay-based mesh reconstructions is one of the most relevant reasons why we estimate 3D edges, we also compared the two 3D meshes reconstructed through the algorithm described in [19] from the OpenMVG points, from Line3D++ [8] and from the points sampled from the 3D edges. As suggested in [21] we compare depth map generated by the reconstructed and the ground truth meshes from the central camera of the sequence of each dataset. Table 1 shows that our algorithm extends to a multi-view stereo setting the hypothesis suggested in [19]: a Delaunay-based reconstruction significantly improves whenever we adopt 3D points belonging to 3D real world edges. Indeed, the meshes estimated from 3D edges points are considerably more accurate than the meshes computed with only SfM points (see Figure 6). Moreover Table 2 shows that in general, in the context of 3D reconstruction our approach generates a point clouds that induce more accurate mesh with respect to the mesh reconstructed on the point cloud generated by Line3D++. Execution times range from a minimum of 4 minutes (DTU-118), to a maximum of 30 (DTU-023), and average at of 13 minutes for the considered datasets.

6. Conclusion and Future Works

In this paper we proposed a novel method to estimate 3D edges and introduced EdgeGraph3D: a system able to recover 3D edges in a scene observed in a set of views. The source code that implements the proposed system is also made available at https://github.com/abignoli/EdgeGraph3D. While existing methods rely on video sequences and estimate only straight edges, our algorithm is able to recover straight and curved edges from an unordered set of images. We represent the image edges as edge-graphs and we match them according to epipolar and spatial constraints. We also showed how Delaunay-based 3D reconstruction improves when built upon points sampled from reconstructed 3D edges. As a future work we plan to integrate the 3D edges into the bundle adjustment process and to embed the recovered 3D edges into the reconstruction algorithm exploiting the Constrained 3D Delaunay triangulation.

Acknowledgements

This work has been partially supported by the “Cloud4Drones” project founded by EIT Digital.

References

[1] A. Ansar and K. Daniilidis. Linear pose estimation from points or lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):578–589, 2003.

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[3] A. Bódis-Szomorú, H. Riemenschneider, and L. Van Gool. Efficient edge-aware surface mesh reconstruction for urban

scenes. *Computer Vision and Image Understanding*, 157:3–24, 2017.

[4] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[5] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[6] E. Eade and T. Drummond. Edge landmarks in monocular slam. *Image and Vision Computing*, 27(5):588–596, 2009.

[7] D. Girardeau-Montaut. Cloud compare, (last access mar, 22 2015).

[8] M. Hofer, M. Maurer, and H. Bischof. Line3d: Efficient 3d scene abstraction for the built environment. In *German Conference on Pattern Recognition*, pages 237–248. Springer, 2015.

[9] C. Hoppe, M. Klopschitz, M. Donoser, and H. Bischof. Incremental surface extraction from sparse structure-from-motion point clouds. *Proc. BMVC*, 2013.

[10] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413. IEEE, 2014.

[11] S. Li, S. Y. Siu, T. Fang, and L. Quan. Efficient multi-view surface refinement with adaptive resolution control. In *European Conference on Computer Vision*, pages 349–364. Springer, 2016.

[12] Z. Li, K. Wang, W. Zuo, D. Meng, and L. Zhang. Detail-preserving and content-aware variational multi-view stereo reconstruction. *arXiv preprint arXiv:1505.00389*, 2015.

[13] V. Litvinov and M. Lhuillier. Incremental solid modeling from sparse and omnidirectional structure-from-motion data. In *BMVC*, 2013.

[14] P. Meer and B. Georgescu. Edge detection with embedded confidence. *IEEE Transactions on pattern analysis and machine intelligence*, 23(12):1351–1365, 2001.

[15] P. Moulon, P. Monasse, and R. Marlet. Adaptive structure from motion with a contrario model estimation. In *Computer Vision–ACCV 2012*, pages 257–270. Springer, 2013.

[16] P. Moulon, P. Monasse, R. Marlet, and Others. Openmvg. an open multiple view geometry library. `https://github.com/openMVG/openMVG`.

[17] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Generic and real-time structure from motion. In *BMVC*, pages 1–10, 2007.

[18] A. Romanoni, M. Ciccone, F. Visin, and M. Matteucci. Multi-view stereo with single-view semantic mesh refinement. In *IEEE International Conference on Computer Vision (ICCV)*, pages 706–715, 2017.

[19] A. Romanoni and M. Matteucci. Incremental reconstruction of urban environments by edge-points delaunay triangulation. In *Intelligent Robots and Systems, 2015. IROS 2015. IEEE/RSJ International Conference on*. IEEE, 2015.

[20] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. *ACM transactions on graphics (TOG)*, 25(3):835–846, 2006.

[21] C. Strecha, W. von Hansen, L. Van Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[22] C. J. Taylor and D. J. Kriegman. Structure and motion from line segments in multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1021–1032, 1995.

[23] Y. Tian, M. Gerke, G. Vosselman, and Q. Zhu. Automatic edge matching across an image sequence based on reliable points. *the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science Beijing*, 37:657–662, 2008.

[24] E. Tola, C. Strecha, and P. Fua. Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Machine Vision and Applications*, 23(5):903–920, 2012.

[25] R. G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. Lsd: a line segment detector. *Image Processing On Line*, 2:35–55, 2012.

[26] H. H. Vu, P. Labatut, J.-P. Pons, and R. Keriven. High accuracy and visibility-consistent dense multiview stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(5):889–901, 2012.

[27] C. Wu. Towards linear-time incremental structure from motion. In *3D Vision-3DV 2013, 2013 International Conference on*, pages 127–134. IEEE, 2013.

[28] C. Zhang and E. P. Baltsavias. *Knowledge-based image analysis for 3D edge extraction and road reconstruction*. Institute of Geodesy and Photogrammetry, ETH-Hoenggerberg, 2000.